Journal of Korea Multimedia Society Vol. 20, No. 8, August 2017(pp. 1406-1420)
https://doi.org/10.9717/kmms.2017.20.8.1406

# Optimized Neural Network Weights and Biases Using Particle Swarm Optimization Algorithm for Prediction Applications

Ezat Ahmadzadeh[†], Jieun Lee[††], Inkyu Moon[†††]

## ABSTRACT

Artificial neural networks (ANNs) play an important role in the fields of function approximation, prediction, and classification. ANN performance is critically dependent on the input parameters, including the number of neurons in each layer, and the optimal values of weights and biases assigned to each neuron. In this study, we apply the particle swarm optimization method, a popular optimization algorithm for determining the optimal values of weights and biases for every neuron in different layers of the ANN. Several regression models, including general linear regression, Fourier regression, smoothing spline, and polynomial regression, are conducted to evaluate the proposed method's prediction power compared to multiple linear regression (MLR) methods. In addition, residual analysis is conducted to evaluate the optimized ANN accuracy for both training and test datasets. The experimental results demonstrate that the proposed method can effectively determine optimal values for neuron weights and biases, and high accuracy results are obtained for prediction applications. Evaluations of the proposed method reveal that it can be used for prediction and estimation purposes, with a high accuracy ratio, and the designed model provides a reliable technique for optimization. The simulation results show that the optimized ANN exhibits superior performance to MLR for prediction purposes.

**Key words:** Artificial Neural Network, Fourier Regression, Multiple Linear Regression, Particle Swarm Optimization, Polynomial Regression, Residual Analysis, Smoothing Spline.

## 1. INTRODUCTION

Artificial neural networks (ANNs) provide a fast and reliable method for solving classification, recognition, clustering, function approximation, and prediction problems [1, 2]. An ANN with multiple layers can be simply applied to many mathematical problems to solve complex non-linear equations, using convenient weights and activation functions [3]. In this type of ANN, the input layer is for applying input data, while the hidden layer and output layers obtain final answers to the question. Every layer may consist of a different number of neurons, each of which has a weight and bias value assigned to it. Generally, artificial ANN serves as an accurate modeling tool; however, to obtain the desired accuracy precision in various circumstances, the ANN must be trained under different possible con-

※ Corresponding Author : Jieun Lee, Address: (501-759) Department of Computer Engineering, Chosun University, 309 Pilmun-daero, Dong-gu, Gwangju, 501-759 South Korea, TEL : +82-62-230-6033, FAX : +82-62-230-6033, E-mail : jieunlee@chosun.ac.kr
Receipt date : May 12, 2017, Revision date : Jul. 1, 2017
Approval date : Jul. 4, 2017
[†] Department of Computer Engineering, Chosun University, 309 Pilmun-daero, Dong-gu, Gwangju, 501-759 South Korea (E-mail : ezat.ahmadzade@gmail.com)
[††] Department of Computer Engineering, Chosun University, 309 Pilmun-daero, Dong-gu, Gwangju, 501-759 South Korea
[†††] Department of Computer Engineering, Chosun University, 309 Pilmun-daero, Dong-gu, Gwangju, 501-759 South Korea
(E-mail : inkyu.moon@chosun.ac.kr)
※ This work was supported by research funds from Chosun University in 2011.

ditions [4,5]. During the ANN training phase, neuron weights and biases are updated several times until optimal values are obtained for each neuron, to obtain the most accurate output. The backpropagation learning algorithm is the most popular ANN learning method, which uses gradient descent learning method. However, the problem is that gradient-based algorithms may trap in the local minimum, whereas we need to determine the global minimum to obtain the best result [6]. The gradient-based ANN learning method can be replaced with optimization algorithms. Since optimization methods try to find global minimum and prevent from falling into local minimum, in this study we try to replace back propagation learning method with an optimization method. We apply the particle swarm optimization (PSO) to determine global optimal values for the ANN weights and biases. The PSO algorithm concept was first introduced by Kennedy and Eberhard in 1995 [8]. PSO is an evolutionary algorithm inspired by nature, in the form of the social behavior of animals, such as bird and fish processions. As the name implies, PSO is an optimization method based on particle mass. The basis of the PSO algorithm's development is that all possible answers in an N-dimensional search space are considered as individual particles, randomly distributed in all search spaces at the beginning of the search process [9-11]. The particles move in an N-dimensional space, and their route within the search space changes depending on the particle's past experiences as well as that of its neighbors. The PSO algorithm employs a simple principle and exhibits acceptable performance; therefore, it is a convenient method for determining the optimal weights and biases of neurons in different ANN layers, with a very high probability and convergence rate [12-16]. In this study, we propose a combined technique of ANN weights and biases optimization with PSO, and evaluate its performance for prediction application. The combined technique is a smart optimization approach with many other applications in computer network optimization problems, including sensor network topology optimization and routing protocols, gateway development, robotics, and remote sensing and multi-sensor detection. We furthermore evaluate the prediction power of the proposed method against several multiple linear regression (MLR) models.

In this study, a backpropagation multi-layer ANN is selected for optimization purposes. We selected the hyperbolic tangent sigmoid transfer function (tansig) as the activation function for all layers. The conventional gradient-descent-based ANN training method was replaced by the PSO method to determine global best values for ANN weights and biases. The network mean square error (MSE) was considered as the main criterion for the training process. We trained the network for limited iterations, when there is no significant change in network MSE converge, then the MSE value of optimized neural network for training and test stages are measured. Several MLR analyses were conducted to measure the proposed method's accuracy for prediction application against that of MLR. Finally, we conducted a residual analysis of the training and test data to measure the differences between network-predicted and actual values for each data point.

The remainder of this paper is organized as follows. In section 2, we discuss certain related works. In section 3, we explain the multi-layer ANN, and its training steps and behavior. In section 4, the PSO algorithm and its properties are discussed in depth. We explain our proposed method for determining optimal ANN weights and biases values using the PSO algorithm in section 5. The experimental results of the MLR models are outlined in section 6. Finally, we provide our conclusions in section 7.

## 2. THEORY

### 2.1 Related works

The PSO algorithm is used to find optimal cussing matched filter parameters for retina vessel segmentation [17]. Two retina data sets are applied to this approach, and the obtained results are compared with those of other methods. The proposed method exhibits superior results to other methods [17]. Improvement of ANN training for indoor environment prediction and various training methods are applied to test the proposed ANN system. The proposed method's experimental results are compared with one another, and it is found that PSO and parallel genetic algorithm (GA) show superior performance for training the ANN [18]. Suresha et al. have applied PSO to ANN, and its performance is compared to the backpropagation ANN training method [19]. A comparison of the proposed method and the backpropagation method is conducted by means of error convergence and accuracy.

An investigation is conducted into induction motor control speed by a hybrid of the PSO-ANFIS and PSO application in designing the fuzzy controller of the induction motor [20]; the parameters applied to the designed models are speed and torque. In addition, results are compared with the backpropagation method, and demonstrate that the proposed method exhibits superior performance. Amarash et al. have used PSO for structuring and training ANNs, and performance evaluation has been conducted for network performance. In their experiment, several standard data sets were used to evaluate the proposed method, and results were compared [21].

### 2.2 Multi-layer neural network architecture

ANN concepts were introduced following the recognition of the ability of human brain nerve cells [25]. In 1986, Rumelhart introduced a multi-layer ANN known as a multi-layer perceptron (MLP). The basic structure of the MLP is shown in Fig. 1. The output of each neuron is calculated by the following equation.

$$y_i = f\left[\sum_{i=1}^{n}(w_{ij}.x_i + b_i)\right], \quad (1)$$

where $i$ and $j$ are index numbers of the corresponding cell of the weight matrix in each layer, $w$ is a neuron's weight, $b$ is the bias value of each neuron, and $x$ is the input data value. Gradient descent is the most common ANN training algorithms. Based on measurement of output error, it calculates the gradient of this error, and updates the ANN

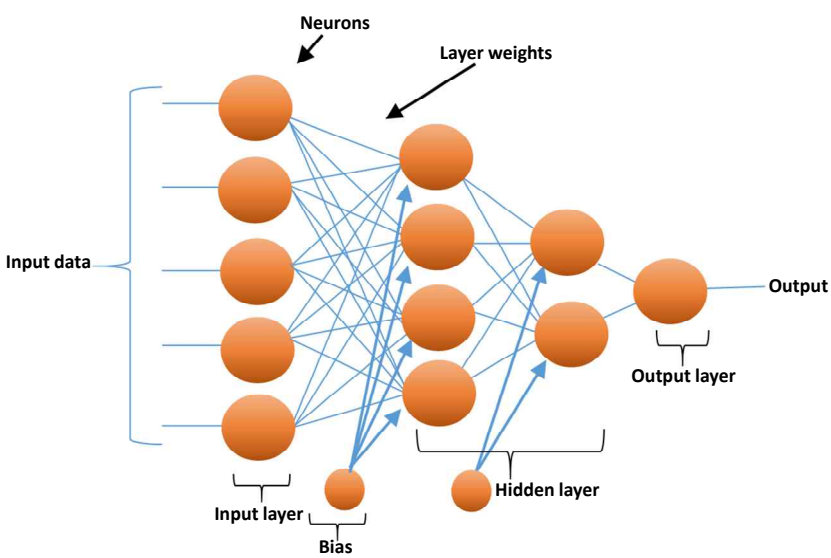


Fig. 1. Schematic diagram of multi-layer ANN structure with one hidden layer.

weights and biases. Learning rate is a major factor in determining how a network is trained. However, the main problem with the gradient descent algorithm is that it may be trapped in local minima [15]. To avoid this, decrement of oscillation distance is used, and to decrease the sensitivity of the network to fast changes in the error surface, ANN weights are changed regarding previous weights changes by adding a momentum coefficient. The network MSE, which indicates differences between real and network output, is obtained with the following equation.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x-\bar{x}), \tag{2}$$

where $x$ is the actual data value, $\bar{x}$ is the value predicted by the network, and $n$ is the total data number.

### 2.3 Particle swarm optimization algorithm

PSO is an evolutionary algorithm inspired by the social behavior of bird flocks and fish aggregation populations [9], and is widely used in asymmetric optimization problems to optimize an objective function using a population-based search. In this method, the population is initialized with a number of random solutions, which are called particles, as with a bird in a flock. These particles can search freely in an N-dimensional search space and are initially assigned a random value. Every individual member of a flock is considered as a solution to the search problem, and each particle has a fitness value depending on the problem to be solved. The most appropriate position of each particle is known as P.best, which is the particle's best fitness value up to the current time, while G.best is the particle with the highest fitness value among the entire population. During the flight, every particle's position and velocity is updated, and as long as they have not reached the global minimum point, the operation is repeated. The target point depends on the type of problem. The velocity of each particle is updated by means of the following equation [9].

$$\begin{aligned} v_{id}^{(t+1)} &= wv_{id}^{(t)} + u[0,1]\varphi_1(p_{id}^{(t)} + x_{id}^{(t)}) \\ &\quad + u[0,1]\varphi_2(p_{id}^{(t)} + x_{id}^{(t)}), \end{aligned} \tag{3}$$

where $u[0,1]$ is a uniform random distribution of particles, $t$ is a time index, and $j_1$ and $j_2$ are weights trading off the impact of the local and global minimum on the particle's total velocity [16, 17]. The particle's position is updated using the velocity equation and the following equation:

$$x_{id}^{(t+1)} = x_{id}^{(t)} + v_{id}^{(t+1)}, \tag{4}$$

where $x_{id}^{(t+1)}$ is the particle's next position $x_{id}^{(t)}$ is its current position, and $v_{id}^{(t+1)}$ is the velocity.

As shown in Fig. 2, in the initial state of PSO algorithm, Fig. 2(a), several random answers denoted as particles are chosen from all possible answers in the N-dimensional search space. Fitness value for all particles are calculated and particle with best fitness value is denoted as G.best which means this particle is more close to the global minimum answer than other particles. In the second step all particle tend to move toward G.best particle using the velocity EQ(3) and update their position using EQ(4) as shown in Fig. 2(b) and fitness value for all particles will be updated and G.best particle will be determined according to new fitness values due to the new positions of particles. This process will be continued Fig. 2(c) and particles will gradually get closer to the global minimum point by updating their position in each step. Finally the process will be ended when there is no improvement in fitness of G.best particle Fig. 2(d) and final answer is denoted as G.best particle.

## 3. PROPOSED METHOD

In our proposed method, the traditional gradient-based ANN training method, which is normally used to determine the optimal values of neuron weights and biases, is replaced with PSO. Every particle in this method is a matrix with the size of all network neurons' weights and biases matrix. With every iteration, the fitness value for

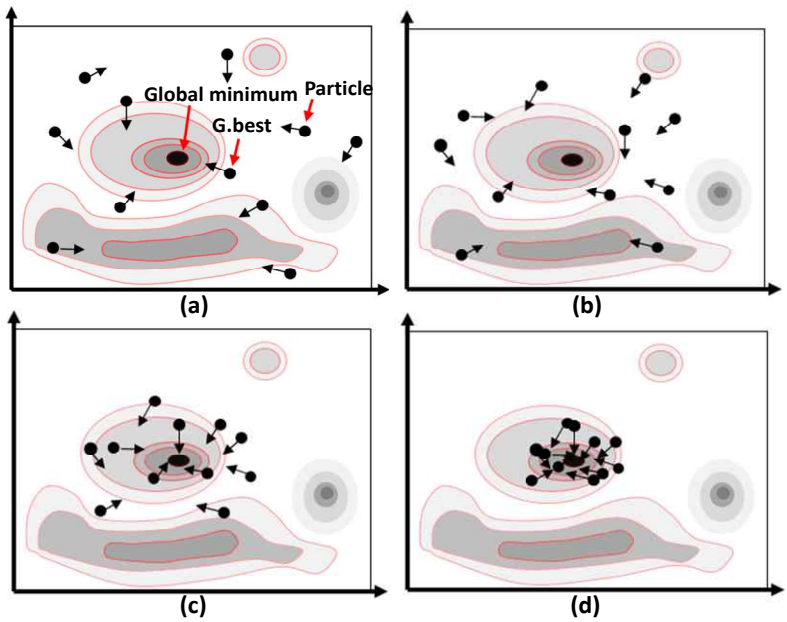


Fig. 2. Initial state of PSO: (a) particles randomly distributed in an N-dimensional search space, (b) particles update their position and move towards the global minimum, (c) particles gradually move closer to the global minimum by updating their position with each iteration, and (d) particles reach the global minimum point.

each particle is calculated, and the particle with highest fitness value is known as G.best. The network is then simulated with the G.best particle, which contains the most optimal weights and biases. Following network simulation with the optimal weights and biases, the network MSE value is calculated. With every iteration, each particle's position is updated using the PSO method, and the network is simulated using the new G.best value, until the stop criterion is satisfied. We considered the network MSE as the cost function for PSO to be minimized with each iteration. The algorithm of the proposed method proceeds according to the following steps: 1) Network topology determination, 2) Layer weights and biases initialization, 3) Random distribution of particles, 4) Calculation of fitness value for each individual particle, 5) G.best and P.best determination, 6) Simulation of network with G.best value, 7) Updating of each particle's velocity using P.best and G.best by means of PSO, 8) Updating of each particle's position, 9) Calculation of network MSE, 10) If termination condition is satisfied then exit, else 11) Go to 4.

Every particle gradually approaches the global minimum by updating its position regarding the P.best and G.best particle. The algorithm repeats for a certain number of iterations or until a pre-defined MSE value condition is satisfied. Once the algorithm ends, the G.best value is considered as the most optimal solution to the optimization problem. Fig. 3 shows the PSO method for determining the global minimum point using several particles searching in an N-dimensional search space. As can be seen in Fig. 3, particles are initially randomly distributed in the N-dimensional search space. The particle closest to the global minimum point is known as G.best, with the highest fitness value among all the particles. In every iteration, the particles update their position in order to move towards the global minimum regarding the G.best particle and their own experience.

### 3.1 Data preparation

We implemented the proposed method in

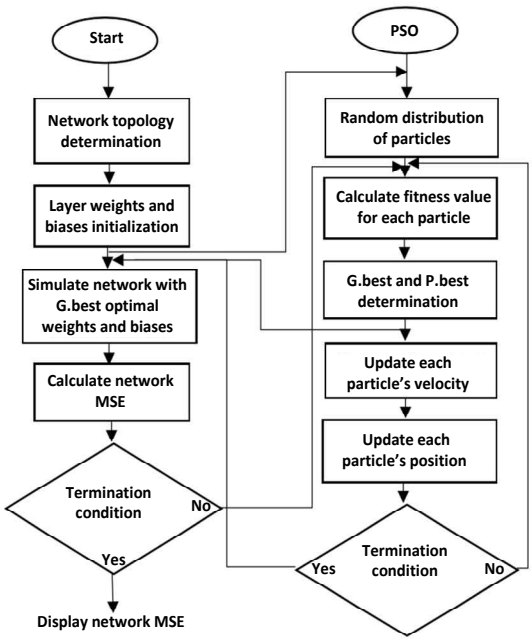


Fig. 3. Flow chart of the proposed ANN weights and biases opti-mization method.

MATLAB. We divided the total data set of 506 observations into two main groups, namely the training set (80%) and test set (20%). Each data observation consisted of 8 measurements. In order to reduce database redundancy, improve data integrity, and increase the speed of the proposed method, we applied normalization to the data set; therefore, our data set was consistently unambiguous. During the normalization stage, we decreased the data observation features range to between -1 and 1. The proposed model was trained using 404 randomly selected data samples, while the remaining 102 samples were used for test data. The PSO initial parameters and data distribution for the training and test phase are displayed in Table 1. The normalization method carried out in this experiment used the following equation:

$$y = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} - 1, \quad (5)$$

where $y$ is the normalized value of data point $x=(x_1,x_2,\cdots,x_n)$. The total data distribution and initial parameters of PSO are shown in Table 1.

## 4. RESULTS AND ANALYSIS

In this study, all experiments were performed in the following environment: 1) Computer: Intel® CoreTM i5-2500, 2) CPU: 3.30 GHz, 3) RAM: 4 GB, 4) OS: Windows 7, and 5) MATLAB: R2016a. We considered network MSE as the cost function for PSO. Therefore, particles attempt to determine the optimal values of weights and biases in order to decrease the cost value, and move towards the minimum cost value. In every iteration, the network is simulated with the G.best particle value, which consists of the most optimal weights and biases among all particles, and network MSE is measured. As seen in Fig. 4, the network MSE trend is downward in every iteration, which means particles move towards finding the optimal values for weights and biases. Fig. 4 shows network MSE for each iteration of network simulation with G. best particle values. We decided to limit the search iteration to 30 epochs when there was no significant change in network MSE.

Fig. 4 illustrates the neural network MSE changes per iteration during the training phase. Initially, weights and biases are randomly assigned to ANN neurons. During the proposed method's training process, particles attempt to determine optimal weights and biases in the N-dimensional search space. Particles update their position in order to approach the global minimum and decrease the network MSE value. As shown in Table 2, as

Table 1. Total data distribution for training and test data, and PSO parameters

| Total data | Training data (%) | Test data (%) | Total training data | Total test data | PSO initial population | Max iteration |
|---|---|---|---|---|---|---|
| 506´8 | 80 | 20 | 404 | 102 | 200 | 30 |

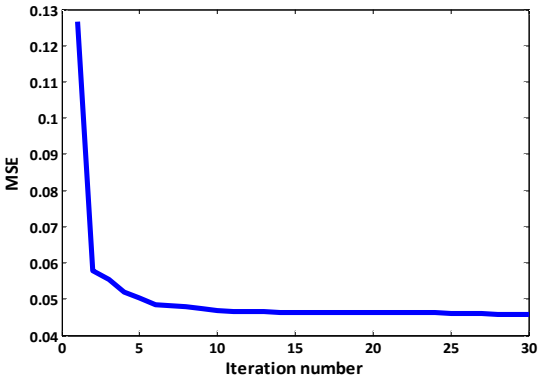


Fig. 4. Neural network MSE changes per iteration during training phase.

the initial step of the ANN training phase, layer weights are randomly assigned to each neuron. Table 2 displays the initial values of layer weights.

In every iteration, the network is simulated with the G.best matrix weights and biases values. Finally, after 30 iterations of PSO search for the global minimum cost, the process ends and the network is simulated with the final optimal weights and biases values obtained from the G.best particle using PSO, and the final training and test phase costs are obtained. Table 3 shows the G.best costs for different iterations, which is considered as the network MSE value.

As shown in Table 3, the G.best particle moves towards the minimum cost in every iteration. After reaching the maximum iteration, final G.best optimal layer weights and biases values are obtained and the network is simulated with the final optimal values. Network error is calculated through following equation. Then we can obtain accuracy using error rate -100.

$$y=\left(\sqrt{MSE}\right)/2*100 \tag{6}$$

Table 4 shows the network accuracy parameters obtained from proposed method. The error rate per test and train stages is calculated using Eq. (5) and respectively network accuracy is obtained using error rate - 100. Table 5 displays the final optimal layer weights obtained using the PSO algorithm.

As shown in Table 3, during the ANN training phase, the G.best particle changes the initial layer weights and biases and determines their optimal values, and finally, the network is simulated with

Table 2. Initial matrix of layer weights generated by ANN

| Neuron number | Layer weight | Layer weight | Layer weight | Layer weight | Layer weight | Layer weight | Layer weight |
|---|---|---|---|---|---|---|---|
| Neuron #1 | 0.9534 | 0.3193 | -0.6098 | -0.1119 | 0.9383 | -0.8805 | -0.2310 |
| Neuron #2 | 0.3823 | 0.1153 | -1.0353 | -0.3524 | 1.2416 | 0.0667 | 0.4502 |
| Neuron #3 | -0.0555 | 0.5974 | -1.0548 | 0.7121 | -0.8323 | 0.0820 | 0.6520 |
| Neuron #4 | 0.3514 | 0.0567 | -1.1007 | 0.3226 | -0.9108 | 0.9109 | 0.0506 |
| Neuron #5 | 0.1260 | 1.3915 | -0.2689 | 0.7666 | 0.5532 | -0.0427 | -0.4292 |

Table 3. G.best particle trend moving towards the minimum cost in each iteration, which is considered as the network MSE

| Iteration | 1 | 3 | 4 | 7 | 8 | 15 | 30 |
|---|---|---|---|---|---|---|---|
| G.best cost | 0.1265 | 0.05547 | 0.05199 | 0.04821 | 0.04791 | 0.04643 | 0.04576 |

Table 4. Network accuracy measurement for training and test data

| MSE training | MSE training | Train Error rate | Test Error rate | Network accuracy for train data | Network accuracy for test data |
|---|---|---|---|---|---|
| 0.0075 | 0.0102 | 4.33 % | 5.04 % | 95.67 % | 94.96 % |

Table 5. Final optimal values of layer weight matrix obtained from G.best particle of PSO

| Neuron number | Layer weight | Layer weight | Layer weight | Layer weight | Layer weight | Layer weight | Layer weight |
|---|---|---|---|---|---|---|---|
| Neuron #1 | 0.2461 | -0.9811 | -0.3948 | 0.3034 | 0.6789 | 0.9831 | 0.0705 |
| Neuron #2 | 0.9867 | 0.7697 | 0.1352 | -0.3458 | 0.3778 | -0.3857 | 0.1884 |
| Neuron #3 | -0.3341 | -0.5844 | 0.4213 | 0.4915 | 0.3391 | 0.1214 | 0.9962 |
| Neuron #4 | 0.1478 | -0.2819 | -0.2479 | 0.2439 | 0.4026 | -0.4116 | -0.9966 |
| Neuron #5 | -0.5564 | 0.6831 | 0.2412 | 0.2700 | 0.5795 | -0.9902 | 0.4181 |

the optimal values obtained from the G.best particle. We conducted several regression analyses to investigate the proposed method's accuracy for prediction applications.

### 4.1 Multiple linear regression analysis

The most commonly used form of regression analysis is MLR. We implemented MLR models in order to illustrate the degree of correlation between the input predictor data and criterion. A regression model is generally used for estimating the level of correlation between a predictor and criterion. The most commonly used model for statistical analysis, including factor and cluster analysis, the discriminant function, and canonical correlation, is the general linear model (GLM). The main purpose of the GLM is to describe an unknown response vector as a linear function of a predictor or known data. The GLM method is used for various analysis purposes, including univariate and multivariate regression variance, time series, and special analysis. Therefore, this model is significant in all research areas. The purpose of data analysis is to understand the accuracy of the proposed method from a different point of view, in comparison with GLM analysis. GLM uses the following equation to draw the best line, which effectively describes data points [26]:

$$y = ax + b, \tag{7}$$

where $y$ is a dimensional vector of the response variable, $x$ is a predictor variable, and $a$, $b$ are coefficient values. Fig. 5 shows the GLM analysis of data points.

A scatter plot of the observed versus predicted values of the optimized ANN for the linear regression model is demonstrated in Fig. 5. It can be seen that the correlation between the predicted and actual data is considerably high. As shown in Fig. 5, the GLM model cannot accommodate all proposed data points. Fig. 5(b) shows the GLM model for the observed and predicted data using the test data set. It is observed that there is a high

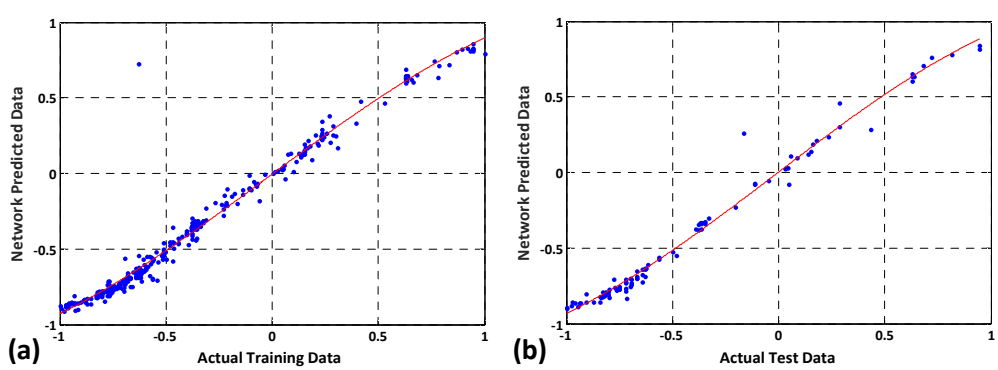


Fig. 5. General linear regression model (GLM) for evaluating predicted data in (a) training phase and (b) test phase.

Table 6. Coefficient parameters of GLM for training and test data

| Coefficient parameters | Training | Test |
|---|---|---|
| $a$ | -0.8069 (-1.072, -0.5421) | -0.8981 (-1.318, -0.4779) |
| $b$ | -0.01166 (-0.02346, .0001481) | -0.008513 (-0.02714, 0.01012) |
| $c$ | 1.16 (-0.02293, 2.344) | 0.853 (-1.015, 2.721) |

level of satisfaction between the actual data points and predicted values with the proposed method during the test stage. The linear model uses the following equation to predict data points:

$$y = a(\sin(x-\pi)) + b(x-10)^2 + c, \quad (8)$$

where $a$, $b$, $c$ are coefficient values, $x$ is input data, and $y$ is the corresponding output value. Table 6 displays the coefficient values of the GLM model for the training and test data set.

### 4.2 Fourier regression analysis

The Fourier curve fitting technique provides powerful measurement with various applications. In the Fourier curve, we can model the predicting data line using a series of sin and cos curves. Fourier analysis has numerous applications in signal processing, physics, probability theory cryptography, geometry, and other areas. In this experiment, we analyzed Fourier curve fitting of predicted values with the proposed method. Fig. 6 shows Fourier curve fitting on the data points [27].

It is clear that there is a close relationship between the actual and predicted data points with the proposed method. The general Fourier fitting model uses the following equation to draw the best fitting line, which can present a sequence of data points perfectly.

$$y = a_0 + a_1 \cos(xw) + b_1 \sin(xw), \quad (9)$$

where $a_0$, $a_1$, $b_1$, and $w$ are coefficient parameters, $x$ is input data, and $y$ is the output point value. From the coefficient parameters in Table 7 and the general Fourier fitting for the test and training data set in Fig. 6, it can be seen that the Fourier fitting model is more suitable for data points than the

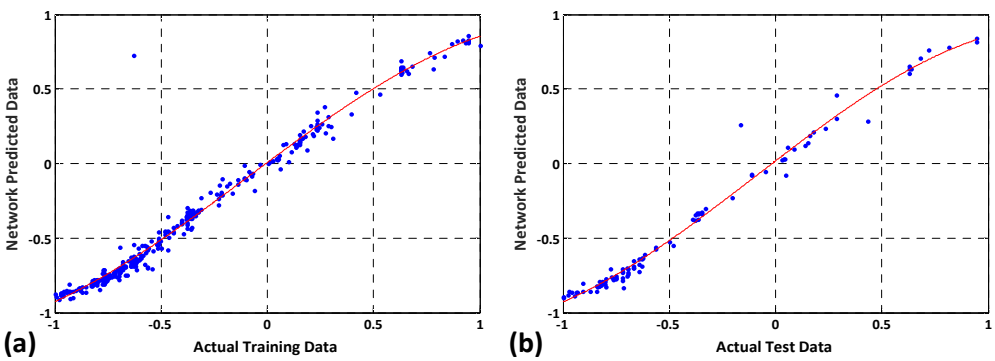


Fig. 6. Fourier curve fit model for train data (a) and test data (b).

Table 7. Coefficient parameters of general Fourier curve fitting model for training and test data

| Coefficient parameters | Training | Test |
|---|---|---|
| $w$ | 0.9986 (0.835, 1.162) | 1.078 (0.8436, 1.313) |
| $a_0$ | -0.08674 (-0.1423, -0.03118) | -0.08476 (-0.1594, -0.01011) |
| $a_1$ | 0.09267 (0.02875, 0.1566) | 0.1012 (0.01379, 0.1886) |
| $b_1$ | 1.059 (0.9175, 1.201) | 1.013 (0.8385, 1.187) |

GLM model.

### 4.3 Fourier regression analysis

Polynomial regression is a form of linear regression that attempts to model the relationship between an independent variable $x$ and dependent variable $y$, and is modeled using the $n$-th degree polynomial of the $x$ variable. The polynomial regression model generally uses the least-squares method, which attempts to minimize the variance of the estimator variable using the Gauss-Markov theorem. In general, polynomial modeling of the dependent variable $y$ as an $n$-th degree uses the following equation [28].

$$y = \sum_{i=0}^{n} a_i x^i + \varepsilon, \tag{10}$$

where $a_0$ to $a_n$ are coefficient parameters, $x$ is a data point value and $y$ is the output value.

Optimized model validation for the polynomial regression model is shown in Fig. 7. In this experiment, we found that 8-degree polynomial regression could fit data better than other degrees. The polynomial regression in this experiment uses the following equation.

$$y = \sum_{i=1}^{8} p_i x^{9-i}, \tag{11}$$

where $p_1,\cdots,p_8$ are coefficient values, $x$ is the input value, and $y$ is the output value. The coefficient parameters of the polynomial equation for the test and training data sets are displayed in Table 8. The coefficient parameter results for the test and training data sets show a high correlation ratio. Correlation analysis aids in determining the most important predictor variables.

### 4.4 Fourier regression analysis

The smoothing spline is a method for fitting a curve to data observation using a spline function, in which data points are considered to be noisy. Fitting of the smooth spline generally consists of two major steps; 1) Drive data number $\bar{f}(x_i)$, $i$=1,2,3,$\cdots..n$, and 2) drive $\bar{f}(x)$, for each data point that uses the following equation:

$$\bar{f}(x) = \sum_{i=1}^{n} \bar{f}(x_i) f_i(x), \tag{12}$$

where $f_i(x)$ is a set of functions based on the spline. The smoothing spline provides a flexible line that attempts to fit the majority of data points underlying the regression function. Furthermore, the smoothing spline can be used for multi-dimensional data. The main problem with smoothing spline regression is that it has a high variance rate. In the case where data points are far from dense of data, regression spline variance gets worse [29]. In this experiment, we conducted smoothing spline regression for test and train data set.

As seen in Fig. 8, the smoothing spline variance for both the training and test data is relatively high. This indicates that certain data points that are far from the dense data are considered to be noise, and

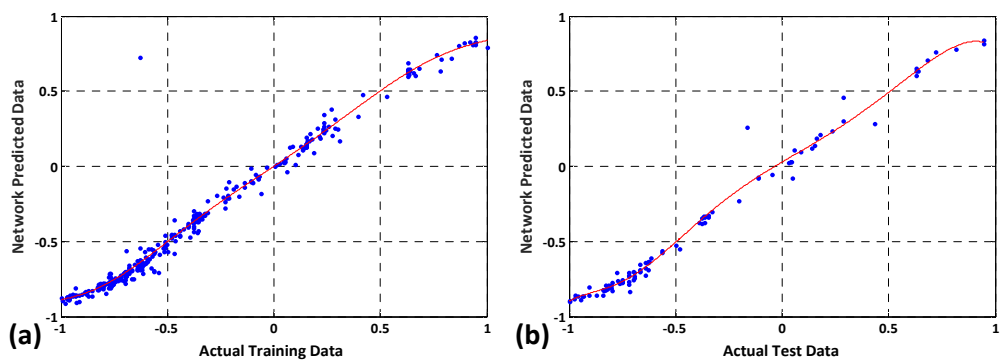


Fig. 7. Polynomial curve fitting model for (a) training data and (b) test data.

Table 8. Polynomial regression coefficient parameters for training and test data

| Coefficient parameters | Training | Test |
|---|---|---|
| $p_1$ | -0.2056 (-2.1, 1.689) | -0.5522 (-3.751, 2.647) |
| $p_2$ | 0.3292 (-0.7337, 1.392) | 0.5311 (-1.697, 2.759) |
| $p_3$ | 0.5447 (-3.037, 4.126) | 0.6116 (-4.983, 6.206) |
| $p_4$ | -0.8477 (-2.49, 0.7942) | -1.525 (-4.876, 1.826) |
| $p_5$ | -0.4445 (-2.69, 1.801) | 0.03973 (-3.177, 3.257) |
| $p_6$ | 0.4386 (-0.3061, 1.183) | 1.006 (-0.4173, 2.43) |
| $p_7$ | 0.08002 (-0.425, 0.585) | -0.1823 (-0.8656, 0.5011) |
| $p_8$ | 0.943 (0.8343, 1.052) | 0.8335 (0.6547, 1.012) |

there is a high amount of spline fluctuation to cover these. To prevent over-fitting and allow for spline shape flexibility, this can be controlled by selecting the number of knots and their location.

A comparison of the actual data observations and predicted data using the proposed ANN model for the training and test data sets is shown graphically in Fig. 9. From the comparison of the actual and predicted data, it is clear that the predicted value is very close to the actual data, and in the majority of cases, they overlap. The consistent agreement between the predicted and actual data points in the training and test data increases the reliability of the proposed model for prediction cases. In addition, it can be concluded from Fig. 9 that the model can be utilized for prediction without requiring more complex experiments. Moreover, an optimized ANN model can be studied for all optimization purposes, which is significant for the present optimization experiment because it shows that optimized ANN weights and biases can be reliably applied for prediction. Fig. 9 shows a comparison between the actual and predicted data points with the proposed method.

### 4. 5 Residual analysis

Residual analysis is a useful technique for analyzing the effectiveness of a designed model. Since an appropriate regression function is extremely important to obtain the most fitted regression lines, verification of the underlying principal is crucial. The difference between the actual data point value, namely the observed value ($y$), and the predicted value ($\hat{y}$) is known as the residual ($R$). Each data point has one residual value showing the difference between the predicted and actual data value [30].

$$R = y - \hat{y}. \tag{13}$$

As can be seen from Fig. 10, the residual plot shows the differences between actual targets and

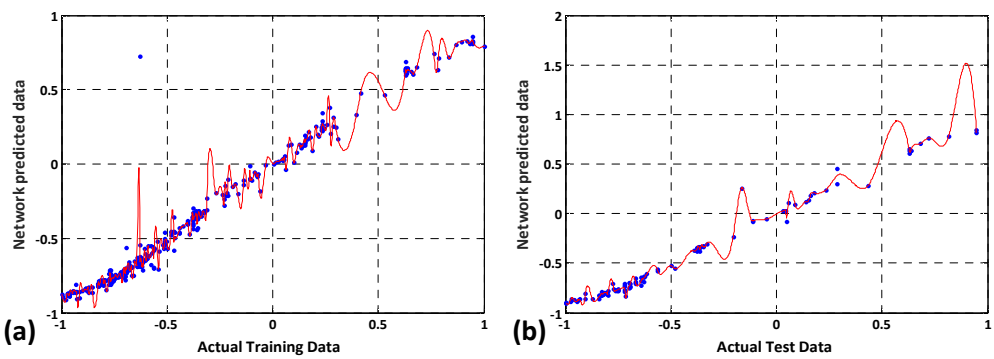


Fig. 8. Smoothing spline model for (a) training data and (b) test data.

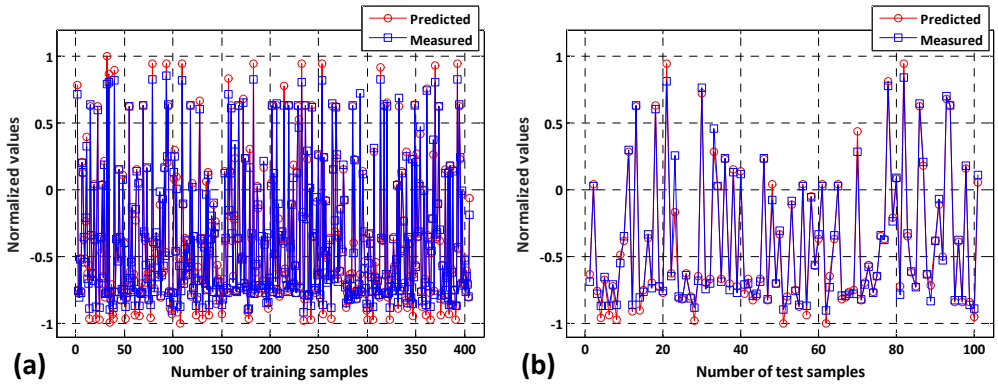


Fig. 9. Comparison of measured individual observation values and predicted values using the proposed method for (a) training data and (b) test data.

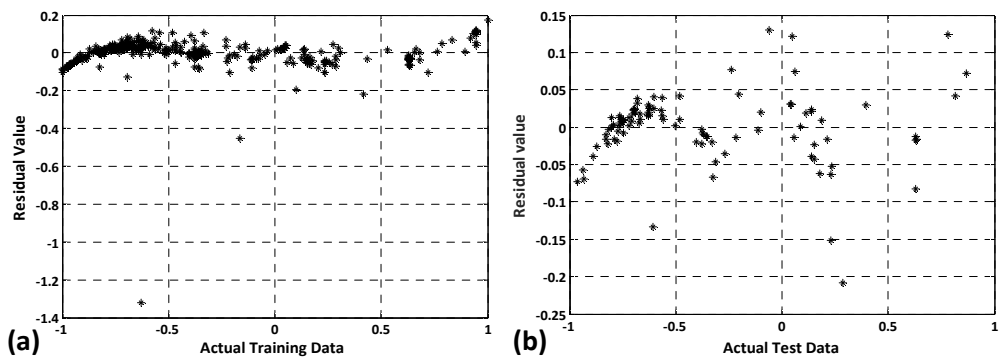


Fig. 10. Residual plot of predicted values using the proposed method for (a) training data and (b) test data.

predicted values using the proposed method, as residual values. The fact that the residual values are close to zero shows that the predicted value is closer to the actual value. A positive residual value demonstrates that the predicted data value is lower than the actual value, while a negative residual value indicates that the predicted data value is higher than the actual value. It is clear from Figure 10 that almost all residual values for the training and test data set were placed around zero, which means that the predicted values were very close to the actual value.

## 5. CONCLUSION

In this paper, we have presented an alternative approach to ANN weights and biases optimization. We applied PSO to MLP ANN to optimize the values of weights and biases parameters, in order to prevent the basic gradient descent method from being trapped in the local minimum, and the proposed method's application was evaluated for prediction application. We compared the validation of the proposed model's prediction power with several MLR models. The proposed method was successfully applied to data sets, and several regression analyses, including GLM, Fourier regression, polynomial regression, and smoothing spline regression, were conducted to evaluate the method's prediction power compared to different MLR models. In addition, residual analysis was conducted to evaluate the proposed method's prediction accuracy for each data point. Network MSE was measured for both the training and test phases of the proposed method.

We demonstrated that ANN neuron weights and biases can be improved using optimization algorithms to obtain the optimal value with minimum training iterations. Furthermore, other optimization algorithms can be applied to the ANN method to determine the optimal number of neurons in each layer or number of layers, and also to determine the optimal activation function for each layer. The experimental results show that the proposed method exhibits superior performance to MLR for prediction applications. Moreover, the potential exists to implement the proposed optimization approach for all types of optimization problems. In terms of the optimized ANN's specific prediction performance, although the results are satisfactory, we can obtain improved performance by increasing the optimization process iterations.

## REFERENCE


[1] A. Maren, C. Harston, and R. Pap, *Handbook of Neural Computing Applications,* Academic Press, Cambridge, 2014.

[2] J. Chae, J. Lim, H. Kim, and J. Lee, "Study on Real-time Gesture Recognition based on Convolutional Neural Network for Game Applications," *Journal of Korea Multimedia Society*, Vol. 20, No. 5, pp. 835-843, 2017.

[3] D. Graupe, *Principles of Artificial Neural Networks*, World Scientific, Singapore, 2013.

[4] F. Gharehchopoghand and E. Ahmadzadeh, "Artificial Neural Network Application in Letters Recognition for Farsi/Arabicmanuscripts," *International Journal of Scientific and Technology Research,* Vol. 1, Issue 8, pp. 49-54, 2012.

[5] E. Ahmadzadeh, W. Azar, and M. Masdari, "A New Approach to Persian and Arabic Handwritten Character Recognition with Hybrid of Artificial Neural Network and Genetic Algorithm," *International Journal of Applied Information Systems,* Vol. 6, pp. 11-15, 2014.

[6] H. Hyungseob and U. Chong, "Neural Network Based Detection of Drowsiness with Eyes Open Using AR Modelling," *IETE Technical Review,* Vol. 33, pp. 518-524, 2016.

[7] K. Duand and M. Swamy, *Particle Swarm Optimization*, Springer International Publishing, Switzerland, 2016.

[8] M. Anantathanavitand and M. Munlin, "Using K-means Radius Particle Swarm Optimization for the Travelling Salesman Problem," *IETE Technical Review,* Vol. 33, Issue 2, pp. 172-180, 2016.

[9] Y. Shi, "Particle Swarm Optimization: Developments, Applications and Resources," *Evolutionary Computation IEEE*, Vol. 1, pp. 81-86, 2001.

[10] H. Tsai, Y. Tyan, Y. Wu, and Y. Lin, "Isolated Particle Swarm Optimization with Particle Migration and Global Best Adoption," *Engineering Optimization*, Vol. 44, Issue 12, pp. 1405-1424, 2012.

[11] J. Kennedy, *Particle Swarm Optimization,* Springer, Switzerland, 2011.

[12] Y. Shi, "Particle Swarm Optimization," *IEEE Connections*, Vol. 2, No.1, pp. 8-13, 2004.

[13] H. Han and U. Chong, "Neural Network Based Detection of Drowsiness with Eyes Open Using AR Modelling," *IETE Technical Review,* Vol. 33, Issue 5, pp. 518-524, 2016.

[14] M. Yaghini, M. Khoshraftar, and M. Fallahi, "A Hybrid Algorithm for Artificial Neural Network Training," *Engineering Applications of Artificial Intelligence*, Vol. 26, Issue 1, pp. 293-301, 2013.

[15] A. Askarzadehand and A. Rezazadeh, "Artificial Neural Network Training Using a New Efficient Optimization Algorithm," *Applied Soft Computting*, Vol. 13, Issue 2, pp. 1206-1213, 2013.

[16] M. Sayed, S. Gharghory, and H. Kamal, "Gain Tuning PI Controllers for Boiler Turbine Unit Using a New Hybridjump PSO," *Journal of*

*Electrical Systems and Information Technology*, Vol 2, Issue 1, pp. 99-110, 2015.

[17] K. Sreejiniand and V. Govindan, "Improved Multiscale Matched Filter for Retina Vessel Segmentation Using PSO Algorithm," *Egyptian Informatics Journal*, Vol. 16, Issue 1, pp. 253-260, 2015.

[18] T. Zhangand and X. You, "Improvement of the Training and Normalization Method of Artificial Neural Network in the Prediction of Indoor Environment," *Proceeding of Engineering*, Vol. 121, pp. 1245-1251, 2015.

[19] A. Suresh, K. Harish, and N. Radhika, "Particle Swarm Optimization over Back Propagation Neural Network," *Proceeding of Computer Science*, Vol. 46, pp. 268-275, 2015.

[20] S. Mahapatra, R. Daniel, D. Dey, and S. Nayak, "Induction Motor Control Using PSO-ANFIS," *Proceeding of Computer Science*, Vol. 48, pp. 753-768, 2015.

[21] A. Sahuand and S. Pattnaik, "Evolving Neuro Structure Using Adaptive PSO and Modified TLBO for Classification," *Proceeding of Computer Science*, Vol. 92, pp. 450-454, 2016.

[22] C. Rajeswari, B. Sathiyabhama, S. Devendiran, and K. Manivannan, "Bearing Fault Diagnosis Using Wavelet Packet Transform, Hybrid PSO and Support Vector Machine," *Proceeding of Engineering*, Vol. 97, pp. 1772-1783, 2014.

[23] K. Anand, B. Barik, K. Tamilmannan, and P. Sathiya, "Artificial Neural Network Modeling Studies to Predict the Friction Welding Process Parameters of Incoloy 800H Joints," *Engineering Science and Technology, an International Journal*, Vol. 18, Issue 3, pp. 394-407, 2015.

[24] M. Khajehand and A. Barkhordar, "Modelling of Solid-phase Tea Waste Extraction for the Removal of Manganese from Food Samples by Using Artificial Neural Network Approach," *Food Chemistry*, Vol. 141, Issue 2, pp. 712-717, 2013.

[25] S. Wang, *Artificial Neural Network*, Springer pp. 81-100, Switzerland, 2003.

[26] J. Cohen, P. Cohen, S. West, and L. Aiken, *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences,* Routledge, United Kingdom, Vol. 17, 2013.

[27] H. Detteand and V. Melas, "Optimal Designs for Estimating Individual Coefficients in Fourier Regression Models," *The Annals of Statistics*, Vol. 31, No. 5, pp. 1669-1692, 2003.

[28] S. Ozdemir, M. Peng, and Y. Xiao, "Polynomial Regression Based Privacy Preserving Data Aggregation for Wireless Sensor Networks," *Wireless Communications and Mobile Computing,* Vol. 15, Issue 4, pp. 615-628, 2015.

[29] C. Gu, *Smoothing Spline ANOVA Models*, Springer Science and Business Media, Vol. 297, Springer, Switzerland, 2013.

[30] R. Cookand and S. Weisberg, *Residuals and Influence in Regression*, Chapman and Hall, New York, 1982.


### Ezat Ahmadzadeh

is a Ph.D. student at the computer engineering department of Chosun University. He received his B.S. degree in Software Engineering in 2011 from Islamic Azad University of Mohabad and M.S. degree in 2014 from Islamic Azad University at the Science and Research Branch of Tehran (West Azarbayjan), respectively. His current research interests include image processing, digital holography, machine vision, and parallel programming.



### Jieun Lee

received his BS degree in computer engineering from Ewha University in 1997, his MS degree from POSTECH in 1999, and his PhD from Seoul National University in 2007. From 1999 to 2002, she worked at LG Electronics Institute of Technology as a research engineer. She is currently an associate professor of the Department of Computer Engineering, Chosun University, Republic of Korea. Her research interests are in geometry processing and computer graphics.



### Inkyu Moon

received his B.S. degree in Electronics Engineering from Sung KyunKwan University in Korea in 1996 and his Ph.D. degree in Electrical and Computer Engineering from the University of Connecticut in the United States in 2007. He joined Chosun University in Korea in 2009 and is currently a Professor at the School of Computer Engineering there. His research interests include digital holography, bio-medical imaging, and optical information processing. Dr. Moon is a member of IEEE, OSA, and SPIE.